An efficient set-based algorithm for variable streaming clustering

Isaac Campos Ardiles[0000−0002−7181−1391], Jared León Malpartida[0000−0001−5697−8188], and Fernando Campos Ardiles[0000−0001−5008−0422]

Department of Informatics
Universidad Nacional de San Antonio Abad del Cusco
Cusco, Peru
{isaac.campos,jared.leon,fernando.campos}@unsaac.edu.pe

Abstract. In this paper, a new algorithm for Data Streaming clustering is proposed, namely the SetClust algorithm. The Data Streaming clustering model focuses on making clustering of the data while it arrives, being useful in many practical applications. The proposed algorithm, unlike other streaming clustering algorithms, is designed to handle cases when there is no available a priori information about the number of clusters to be formed, having as a second objective to discover the best number of clusters needed to represent the points. The SetClust algorithm is based on structures for disjoint-set operations, making the concept of a cluster to be the union of multiple well-formed sets to allow the algorithm to recognize non-spherical patterns even in high dimensional points. This yields to quadratic running time on the number of formed sets. The algorithm itself can be interpreted as an efficient data structure for streaming clustering. Results of the experiments show that the proposed algorithm is highly suitable for clustering quality on well-spread data points.

Keywords: Data streaming clustering · Variable clustering · Disjoint-set operations · Data structures for clustering.

1 Introduction

Clustering is probably the most important unsupervised machine learning problem. While there are several models of clustering, the vast majority of them require working with the entire set of points to be clustered [2,10]. A particular clustering model called *Data Stream Clustering* allows streaming algorithms to cluster points while they arrive. However, the Data Stream Clustering problem focuses on finding clusters given a priori knowledge of the number of clusters [13].

The proposed algorithm (SetClust) tries to solve a very similar problem, the Variable Streaming Clustering. The problem, as in the Data Stream model, handles the case where the data arrives one point at a time; but in this case, there is no a priori knowledge of the number of clusters to be formed. Because of this, the algorithm not only tries to make clustering of the data but also tries

to discover the correct number of clusters *online*, i.e., the number of clusters predicted by the algorithm can considerably change depending on the structure of the data. The algorithm tries to keep the number of predicted clusters as small as possible at any moment.

The experimental part was conducted using synthetic datasets from [7,8,11] aiming to evaluate the performance on well spread clusters and some other closer clusters. The experimental procedure consisted of evaluating the performance of the proposed algorithm and 3 other Data Stream Clustering algorithms on the datasets.

The paper is organized as follows: Section 2 briefly mentions the tool needed to construct the algorithm and some related work on the topic. Then, the SetClust Algorithm is described in Section 3. Section 4 explains how the experiments were performed and the achieved results. Section 5 offers a brief discussion on the algorithm and the relation hyperparameters have with the data. Finally, the conclusions are presented in Section 6.

2 Background

Below, we mention the data structure needed to develop the SetClust algorithm, then the evaluation metric used for testing clustering quality, and finally we mention 3 algorithms that are later used in the experimental comparisons.

Union-Find disjoint sets operations. This data structure [9] allows keeping track of sets of elements storing minimal information about them. The structure allows two kinds of operations: 1) Make a query to know whether two elements are in the same set, and 2) Join two sets into one single set (see [6] for more).

V-mesaure. This metric is an entropy-based measure which explicitly measures how successfully the criteria of homogeneity and completeness have been satisfied. V-measure is computed as the harmonic mean of distinct homogeneity and completeness scores (see [14] for more on this).

Clustream. This Data Streaming Clustering Algorithm is based on micro-cluster structures that store information about a streaming. The algorithm has two phases: an online phase in which the streams are processed and an offline phase in which the clusters are created using k-means (see [1] for more on this).

ClusTree. This Data Streaming Clustering Algorithm is a self-adaptive index structure that keeps the summarized information about the stream. Also, this algorithm adds aging to the data to eliminate unnecessary information (see [12] for more on this).

DenStream This Data Streaming Clustering Algorithm is based on densities, the reason why it can take arbitrary cluster shapes during the execution. This algorithm can easily deal with outliers (see [5] for more on this).

3 The SetClust Algorithm

From now on, the terms "cluster" and "set" will be used independently: we use the term set to denote the minimum unit of grouped information, and the term

cluster to denote a collection of one or more sets joined together by the same label.

At any time, each set is represented by the following information: 1) label of the set, 2) arithmetic mean of the points that belong to the set, 3) standard deviation of the points, 4) sum of element-wise squared points and 5) the number of points. Also, the disjoint-set structure mentioned in Section 2 is built over all the sets. Of course, for each point that arrives, we also must store the label of the set it belongs to. With this information, we define three operations that represent the action to be taken depending on the situation: element aggregation, set linking, and set absorption. The three operations are described below.

Element Aggregation The purpose of the element aggregation operation is to add the information of a point to a set. This is achieved in the following way: given a point $p \in \mathbb{R}^d$, find the set with the nearest mean $\mu_i \in \mathbb{R}^d$ having a standard deviation $\sigma_i \in \mathbb{R}^d$. Then, we take the number $c \in \mathbb{R}^+$ as hyperparameter. The point p is added to the set if the following inequation holds:

$$\|(p - \mu_i)./(\sigma_i \cdot c)\|^2 < 1 \tag{1}$$

Where $./$ is the element-wise division. The intuition behind this condition is that a point is to be added to the set if it lies inside c standard deviations of an ellipsoid with center μ_i. The extension of the ellipsoid in each dimension is given by σ. Given this condition, there are two possible scenarios. First, the point does not belong to the set. In this case, a new set is created having a unique element: the single point. The mean of the set is the point itself, and the standard deviation is initially fixed to σ_{ini} being this last a hyperparameter. At every moment, the standard deviation of each set is at least σ_{ini}. Then, the information about this point is added to the disjoint-set structure. Second, the point belongs to the set. In this case, the information of the set should be updated, i.e., update the mean, standard deviation, and the number of elements[1]. Both calculations have a running time complexity of $O(d)$, being d the dimension the points lie in. The overall complexity of the element aggregation operation is $O(rd)$, being r the number of current formed sets.

Set Absorption The set absorption operation is responsible for converting two close sets into a single one when discovering that one can be included inside another. The operation goes as follows: given two sets with means and standard deviations μ_i, σ_i and μ_j, σ_j respectively and the hyperparameter c, the two sets must become a single one if one set, when extended c times by its standard deviation is included into the other set extended c times by its standard deviation in *all* its components. Formally, the set absorption must be performed if the following inequation holds:

$$|\mu_i - \mu_j| + c \times \sigma_j <^d c \times \sigma_i. \tag{2}$$

[1] These operations should be done online without traversing through all the elements.

Where $<^d$ represents the minor operator on *every* component (see Fig. 1 for a one dimensional example).

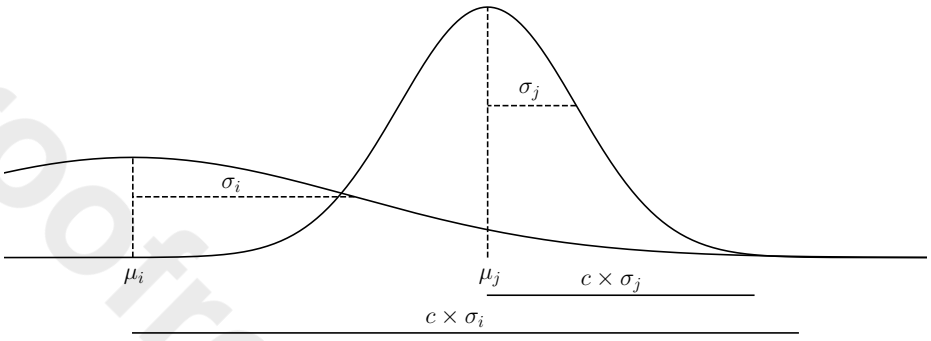

Fig. 1. One dimensional example of two sets satisfying the absorption condition. Set with mean μ_j is inside the acceptance border of set with mean μ_i.

When the absorption condition is met, we proceed to merge both sets recalculating its information to form a new set[2]. The absorption operation is done until no two sets meet the absorption condition. It is possible that after joining two sets the need for making more absorption operations increase since the number of elements that belong to the last modified set also change. The absorption of a set has a running time complexity of $O(r^2d)$. Depending on the implementation, we might need to traverse all sets once for every verification of the condition[3]. This operation has a good effect on the algorithm since every time it is used, the number of sets decreases by one.

Set Linking The goal of the set linking operation is to determine whether two sets are close enough to be considered a single *cluster*. Given two sets with means μ_i, μ_j and standard deviations σ_i, σ_j respectively, we say that both sets are linked if the following inequation holds:

$$\|(\mu_i - \mu_j)./(\sigma_i \cdot c)\|^2 < 1 \tag{3}$$

Here, we use the previously defined hyperparameter c. Intuitively, the condition tests whether the mean of a set satisfies the element aggregation condition for another set.

[2] We need to calculate the new mean, standard deviation and the rest of the information in constant time.

[3] If we take advantage of the fact that only the last formed set can make instability, we can achieve an overall worst-case running time complexity of $O(rd)$.

Once determined if two sets will be linked, a *union* operation of the disjoint-set structure is performed on their labels, meaning that both sets belong to a single cluster. Doing the operation in this way, we allow the algorithm to link sets that create complex cluster forms, not just spherical.

The set linking operation is performed after the set absorption operation because if two sets satisfy the absorption condition, then they satisfy the linking condition with high probability. We restrict linking sets that can be absorbed by any other to reduce the number of clusters. Depending on the implementation, this operation can be performed in almost linear time in the number of sets[4].

It is important to clarify that, whenever a new point arrives, the algorithm tries to perform the three operations (element aggregation, set absorption, and set linking in that order) to the whole structure. The three operations are independently executed one after the other. As mentioned before, the worst-case running time complexity of the operations are: $O(rd)$, $O(rd)$ (at most $O(r)$ times), and $O(\alpha(r)rd)$ respectively. If we assume $\alpha(r)$ to be a small constant, the overall complexity of the algorithm when adding a point is $O(r^2d)$.

When two or more sets are linked, they are considered to belong to the same cluster. This property was specifically chosen for the algorithm as it has an advantage: linked sets can form non-spherical cluster shapes. It is quite easy to see that when two or more sets are joined into a single one, the shape of the final set is "pseudo elliptical". However, when two of these sets link each other, then the resulting shape can be more sophisticated. The linking process can continue until exotic shapes are formed. This last behavior helps the algorithm to achieve better cluster quality since it can make more suitable cluster shapes of the data.

At any moment, we can know if two elements belong to the same cluster with the help of the disjoint-set structure. Also, at any moment, the algorithm can return a list of the labels of the points seen so far. The algorithm can also return the entire structure containing all the information used completely to define the sets and clusters to continue doing clustering if necessary. A high-level pseudocode of the SetClust algorithm is shown in Algorithm 1.

Whenever a new point arrives, an element aggregation operation is performed. This can yield into creating a new set or adding the new point to a previously existing set. In any case, the modified or newly created set is taken as a possible candidate for the absorption operation. If the absorption condition is met, then the absorption operation is performed. The process of verifying the absorption condition and performing the absorption operation continues until no absorption operation can be done. Then, the same set is taken as a candidate for the linking operation, checking for the corresponding condition as in the previous case. If the condition is met, then the set linking operation is performed only once.

[4] As in the previous case, if we take advantage of the fact that only the last formed set can make instability, we can perform this operation in worst-case running time complexity of $O(\alpha(r)rd)$, where α is the inverse Ackerman function. For any practical situation, the function is never greater than 4.

Algorithm 1: The SetClust Algorithm

```
while point p arrives do
    Element Agregation(p)
    Let A be the set where p belongs
    while an absorption can be performed with A do
        Set Absorption(A)
    if a link can be performed with A then
        Set Linking(A)
```

4 Experiments

For the experiments, six synthetic datasets were selected from [7,8,11]. Each dataset contains several points in some specified dimension together with its correct label. Relevant information about the datasets is shown in Table 1.

Table 1. Information about the synthetic datasets used for the experiments

Dataset	Dimension	Number of elements	Number of clusters
Dim512	512	1024	16
Dim1024	1024	1024	16
A1	2	3000	20
A2	2	5250	35
A3	2	7500	50
S1	2	5000	15

The first two datasets contain several Gaussian clusters. Clusters are well separated. The next three are incrementally included; i.e., A1 is included in A2 and A2 is included in A3. These datasets have low dimension and are not so spread as the previous two. The reason for including this type of datasets is to test the performance of the algorithm in an outlier scenario. The last dataset, S1, contain some non-spherical (amorphous) cluster shapes. The purpose of this dataset is to test how algorithms handle this kind of scenarios.

The experimental procedure was performed as follows: for every dataset and algorithm, we performed a grid search to find good hyperparameters. The evaluation of the clustering quality was performed using the V-measure metric [14], which gives a score between -1 and 1. Table 2 shows the results of this procedure. Each numerical value of the table contains the best score seen by the current algorithm on the current dataset. The best score achieved on every dataset (every column) is highlighted in bold.

As showed in the table, the first two datasets were easy for all the algorithms due to its spread nature. The next four datasets were more challenging because of the complex nature of clusters shapes and the addition of some outliers present on them. In those datasets, the proposed algorithm clearly shows superior per-

Table 2. Results of the experiments

	Dim512	Dim1024	A1	A2	A3	S1
SetClust	**1.00**	**1.00**	**0.97**	**0.96**	**0.97**	**0.97**
Clustream	**1.00**	**1.00**	0.79	0.82	0.83	0.80
ClusTree	**1.00**	**1.00**	0.75	0.74	0.56	0.80
DenStream	**1.00**	**1.00**	0.70	0.66	0.61	0.73

formance, showing the highest score on all cases. Numerical experiments were conducted using the Python programming language (Python 3.XX) on a 2.50 GHz Intel Core i7-4710HQ with 12GB of RAM and running Windows 10 version 1809 as operating system. The source code of the SetClust algorithm is available in [4]. For the other algorithms, we used the `MOA` framework for Big Data stream mining [3].

5 Discussion

The selection of hyperparameters σ_{ini} and c can have a deep interpretation when looking at the structure of the dataset used. The parameter σ_{ini} can be interpreted as a first approximation to the standard deviation that a cluster will have, for that, it might be a good practice to choose its value looking at the "separation degree" of the dataset. The parameter c can be seen similarly: its value can be interpreted as a guess of how spread the clusters will be during the streaming. In the same way, looking at the structure of the dataset can help to make a good choice for its value.

The Clustream and ClusTree algorithms make use of the correct number of clusters to be formed. The DenStream algorithm, despite not needing this information, requires tuning a great number of other hyperparameters. The proposed algorithm has exactly two hyperparameters, which makes it suitable for a streaming scenario with semi-unknown information.

6 Conclusions

In this paper, a new algorithm for variable data streaming clustering was proposed, the SetClust algorithm. The algorithm has a disjoint-set operations background and can be used as a data structure with online query support.

A maximum clustering quality was achieved when the clusters are well spread. This case is an optimal scenario for the SetClust algorithm (as well as for the other algorithms). The experiments also show that, for the last four datasets, the proposed algorithm finds better clusters than the other tested algorithms.

In spite of the theoretical efficient based formulation of the algorithm, there are lots of implementation details that give much freedom when implementing it. As future work, we will focus on adapting better data structures to handle redundant information so as to make a more efficient implementation of the algorithm. Also, we plan to create a more sophisticated version of SetClust incorporating

aging to the data. We also plan to perform running time comparisons between the SetClust algorithm and other Data Streaming Clustering algorithms.

References

1. Aggarwal, C.C., Han, J., Wang, J., Yu, P.S.: A framework for clustering evolving data streams. In: Proceedings of the 29th International Conference on Very Large Data Bases. vol. 29, pp. 81–92. VLDB Endowment (2003)
2. Aggarwal, C.C., Reddy, C.K.: Data Clustering: Algorithms and Applications. Chapman & Hall/CRC, 1st edn. (2013)
3. Bifet, A., Holmes, G., Kirkby, R., Pfahringer, B.: Moa: Massive online analysis. Journal of Machine Learning Research **11**, 1601–1604 (2010)
4. Campos, I., Leon, J.: Setclust (2019). https://doi.org/10.5281/zenodo.3270842
5. Cao, F., Ester, M., Qian, W., Zhou, A.: Density-based clustering over an evolving data stream with noise. In: Conference on Data Mining (SIAM '06). pp. 328–339 (2006)
6. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: Introduction to Algorithms, 3rd Edition. MIT Press (2009)
7. Fränti, P., Virmajoki, O.: Iterative shrinking method for clustering problems. Pattern Recognition **39**(5), 761–765 (2006)
8. Fränti, P., Sieranoja, S.: K-means properties on six clustering benchmark datasets (2018)
9. Galler, B.A., Fisher, M.J.: An improved equivalence algorithm. Commun. ACM **7**(5), 301–303 (1964)
10. Jain, A.K., Dubes, R.C.: Algorithms for Clustering Data. Prentice-Hall, Inc. (1988)
11. Kärkkäinen, I., Fränti, P.: Dynamic local search algorithm for the clustering problem. Tech. Rep. A-2, Department of Computer Science, University of Joensuu (2002)
12. Kranen, P., Assent, I., Baldauf, C., Seidl, T.: Self-adaptive anytime stream clustering. In: Ninth IEEE International Conference on Data Mining (ICDM '09). pp. 249–258 (2009)
13. Muthukrishnan, S.: Data streams: Algorithms and applications. Found. Trends Theor. Comput. Sci. **1**(2), 117–236 (2005)
14. Rosenberg, A., Hirschberg, J.: V-measure: A conditional entropy-based external cluster evaluation measure. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning(EMNLP-CoNLL). pp. 410–420 (2007)